Apellidos: **Nombre:** **Matrícula:**

Examen

105000016 - Programación para Sistemas

Grado en Ingeniería Informática

Lenguajes y Sistemas Informáticos e Ingeniería de Software
Facultad de Informática
Universidad Politécnica de Madrid

Curso 2013/2014 - Enero 2014

Normas

- El examen puntúa sobre **12 puntos**.
- La duración total del mismo es de **una hora y cuarto**.
- Se deberá tener el DNI o el carnet de la UPM en lugar visible.
- No olvidar rellenar **apellidos, nombre y número de matrícula** en cada hoja.
- La solución al examen se proporcionará antes de la revisión.
- La fecha prevista de publicación de calificaciones es el **21 de enero**, y se realizará a través del Aula Virtual de la asignatura.
- La revisión del examen tendrá lugar el **23 de enero** a las 12:00 en la sala 2319.

Cuestionario

(1 punto) 1. Suponiendo que las variables `A` y `B` contienen números enteros válidos. ¿Cuál de los siguientes mandatos comprueba si `$A` no es igual a `$B`?

A. `[ $A -eq $B ]` **B. `[ $A -ne $B ]`** C. `[ $A -gt $B ]` D. `[ $A -lt $B ]`

(1 punto) 2. Dado el siguiente mandato Bash (el símbolo `#` indica el *prompt* de la línea de mandatos):

```
# $(echo ls /)
```

Se pide señalar la respuesta correcta:

A. Lista los ficheros y directorios del directorio "/".

B. Saca por la salida estándar el texto "`ls /`".

C. Da un error.

(1 punto) 3. Dado el siguiente script `parametros.sh`:

```
#!/bin/bash
echo $#
```

¿Cuál es el resultado de la siguiente invocación del mismo desde la línea de mandatos?

```
$ ./parametros.sh esto es una prueba
```

Solución:

```
4
```

(1 punto) 4. En el manual de Bash, se puede leer la siguiente descripción sobre la expansión de variables (La construcción es en realidad más compleja pero estas líneas bastan para resolver el ejercicio):

```
${!prefix*}
  Nombres que encajan con prefijo. Expande a los nombre de
  variables que empiezan por prefix separados por espacios
  y ordenados por en orden alfabético.
```

Se pide escribir las tres líneas de la salida estándar resultado de la ejecución de los siguientes mandatos Bash, suponiendo que no hay otras variables definidas que empiecen por "`MIV`":

```
MIVUNO=
MIVDOS=
MIVTRES=
MIVCUATRO=
MIVCINCO=
echo ${!MIV*}
echo ${!MIVC*}
echo ${!MIVU*}
```

Solución:

```
MIVCINCO MIVCUATRO MIVDOS MIVTRES MIVUNO
MIVCINCO MIVCUATRO
MIVUNO
```

(1 punto) 5. **Se pide** escribir el nombre de dos funciones de la librería estándar de C que sirvan para realizar reserva dinámica de memoria.

> **Solución:** `malloc` y `calloc`.

(1 punto) 6. ¿Cuál será la salida del siguiente programa? Justifica la respuesta.

```
void fun(int x)
{
  x = x+5;
}

int main(void)
{
  int x=5;
  fun(x);
  printf("%d\n",x);
  return 0;
}
```

> **Solución:**
>
> 5
>
> Justificación: el paso de parámetro en C se realiza por valor (o copia), no por referencia.

(1 punto) 7. Dado el siguiente código C para la función `g`:

```
void g(int *x)
{
  *x = *x + 1;
}
```

Se pide escribir dos líneas de código C para declarar una varible de tipo **`int`**, asignarle `42` y llamar a la función `g` correctamente (con la intención de incrementar dicha variable en `1`).

> **Solución:**
>
> ```
> int x = 42;
> g(&x);
> ```

(1 punto) 8. **Se pide** escribir la salida del siguiente código:

```
int main(void)
{
  int x=0;
  while(x++ <= 3)
    printf("Hola␣%d\n",x);
  return 0;
}
```

Solución:

```
Hola 1
Hola 2
Hola 3
Hola 4
```

(1 punto) 9. ¿Cuál será el valor de la variable "y" al finalizar la ejecución del siguiente programa C?

```
int main(void)
{
  int x[] = {1, 4, 8, 5, 1, 4};
  int *ptr, y;
  ptr = x + 4;
  y = *ptr - *x;
  return 0;
}
```

Solución:

0

(1 punto) 10. **Se pide** completar el código C de forma que la ejecución del programa resultante imprima en pantalla:

```
0
1
2
```

El código a completar es el siguiente:

```
#include <stdio.h>
#include <stdlib.h>
#define N 3

int main( void ) {
  int i;
  int *p;

  p = (int *) malloc( N * sizeof( int ) );
  for ( i = 0; i < N; i++ ) {
    *(<<a completar>>) = i
  }
  for ( i = 0; i < N; i++ ) {
    printf( "%d\n", p[i]);
  }
  free( p );
  return 0;
}
```

Solución:

p+i

(1 punto) 11. Sea una aplicación en C que se compone de dos ficheros fuente `prog.c` (donde se encuentra la función `main`) y `funciones.c`.

Se pide escribir la llamada al compilador `gcc` que compile los ficheros fuente y genere el ejecutable `prog`. Incluya la opción al compilador adecuada para poder utilizar el depurador.

Solución:

```
gcc -g -Wall -pedantic -ansi -o prog prog.c funciones.c
```

(1 punto) 12. **Se pide** escribir la orden para definir y establecer un punto de parada o breakpoint en el depurador `gdb`.

Solución: `break`